Single Source Shortest Path(SSSP) Dijkstra’s Algorithm

Week-11, Lecture-02

Course Code: CSE221

Course Title: Algorithms

Program: B.Sc. in CSE

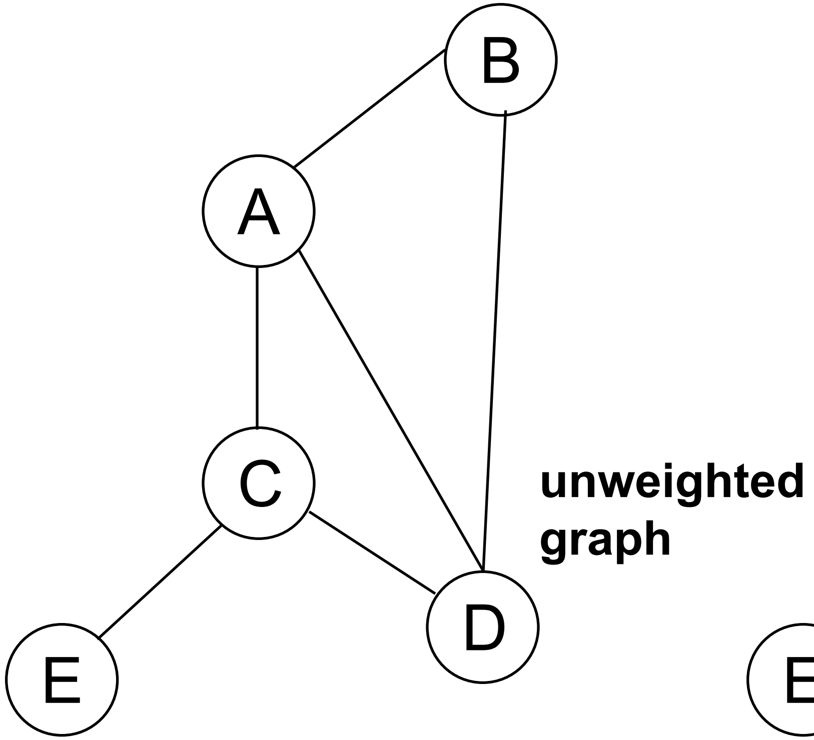
Course Teacher: Tanzina Afroz Rimi

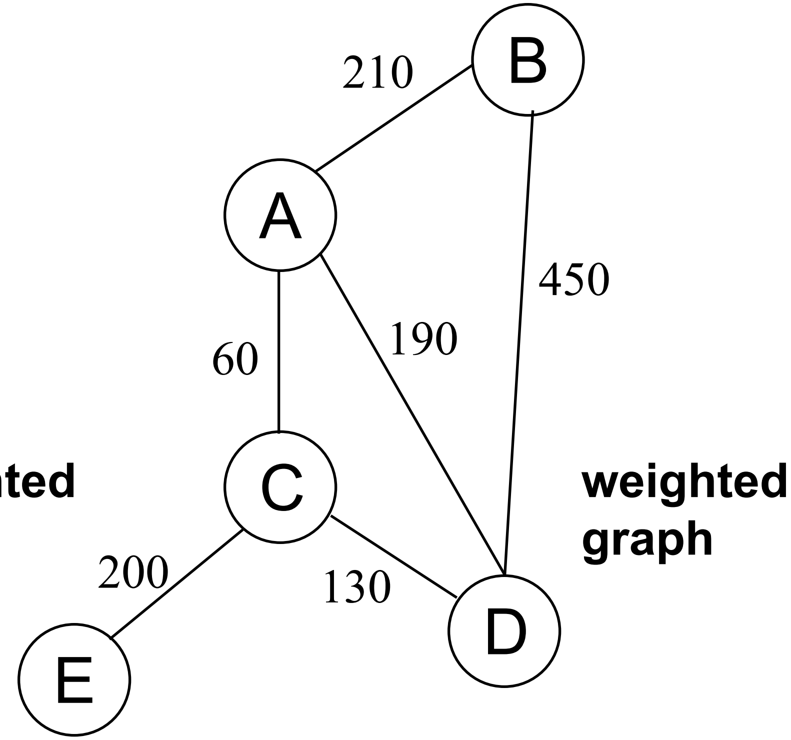
Designation: Lecturer

Email: tanzinaafroz.cse@diu.edu.bd

Shortest Path Problems

- **What is shortest path ?**
  - shortest length between two vertices for an unweighted graph:
  - smallest cost between two vertices *for* a weighted graph:





Shortest Path Problems

- How can we find the shortest route between two points on a map?
- Model the problem as a graph problem:
  - Road map is a weighted graph:

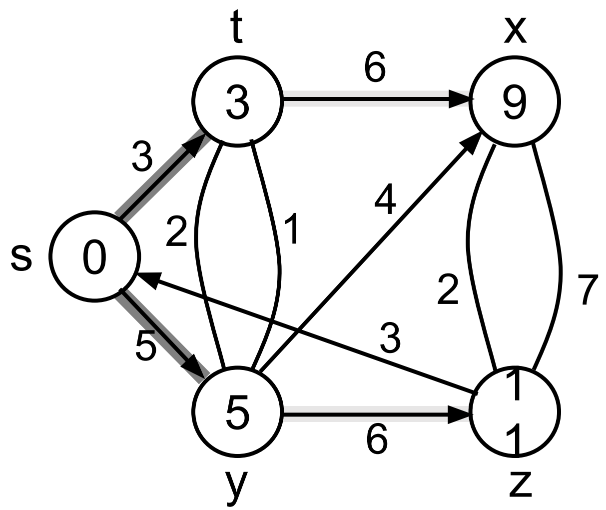
    vertices = cities

    edges = road segments between cities

    edge weights = road distances
  - Goal: find a shortest path between two vertices (cities)

Shortest Path Problems

- **Input:**
  - Directed graph G = (V, E)
  - Weight function w : E ⟶ **R**



- **Weight of path** p = $\langle v_0, v_1, \ldots, v_k \rangle$

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

- **Shortest-path weight** from u to v:

$$\delta(u, v) = \begin{cases} \min \{ w(p) : u \overset{p}{\leadsto} v \} & \text{if there exists a path from } u \text{ to } v \\ \infty & \text{otherwise} \end{cases}$$

- Shortest path u to v is any path p such that w(p) = δ(u, v)

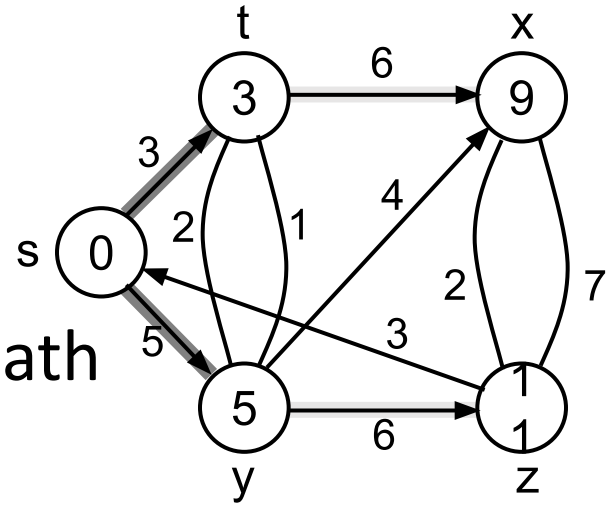
Variants of Shortest Paths

- **Single-source shortest path**
  - $G = (V, E) \Rightarrow$ find a shortest path from a given source vertex s to each vertex $v \in V$
- **Single-destination shortest path**
  - Find a shortest path to a given destination vertex **t** from each vertex v
  - Reverse the direction of each edge $\Rightarrow$ single-source
- **Single-pair shortest path**
  - Find a shortest path from u to v for given vertices u and v
  - Solve the single-source problem
- **All-pairs shortest-paths**
  - Find a shortest path from u to v for every pair of vertices u and v

Shortest-Path Representation

For each vertex $v \in V$:

- $d[v] = \delta(s, v)$: a **shortest-path estimate**
  - Initially, $d[v]=\infty$
  - Reduces as algorithms progress
- $\pi[v]$ = **predecessor** of **v** on a shortest path from s
  - If no predecessor, $\pi[v]$ = NIL
  - π induces a tree—**shortest-path tree**
- Shortest paths & shortest path trees are not unique



Initialization

Alg.: INITIALIZE-SINGLE-SOURCE(V, s)

1. **for** each v $\in$ V

2. **do** d[v] $\leftarrow \infty$

3. π[v] $\leftarrow$ NIL

4. d[s] $\leftarrow$ 0

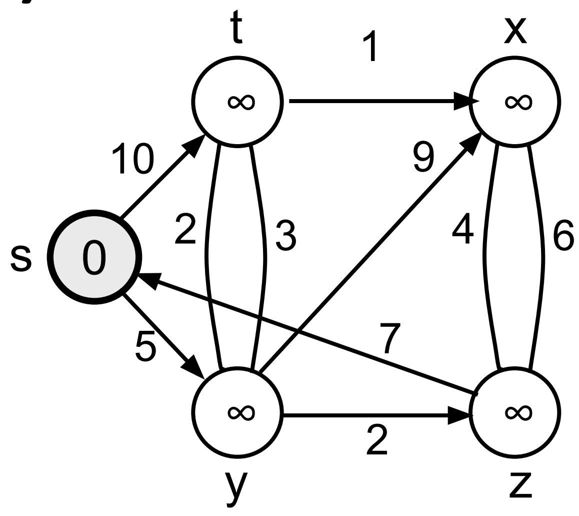
- All the shortest-paths algorithms start with INITIALIZE-SINGLE-SOURCE

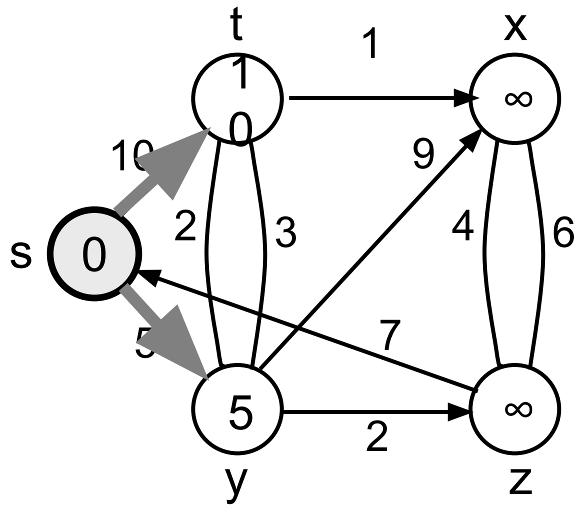
Dijkstra's Algorithm

- Single-source shortest path problem:
  - No negative-weight edges: $w(u, v) > 0 \ \forall \ (u, v) \in E$
- Maintains two sets of vertices:
  - S = vertices whose final shortest-path weights have already been determined
  - Q = vertices in V – S: min-priority queue
    - Keys in Q are estimates of shortest-path weights (d[v])
- Repeatedly select a vertex $u \in V - S$, with the minimum shortest-path estimate d[v]

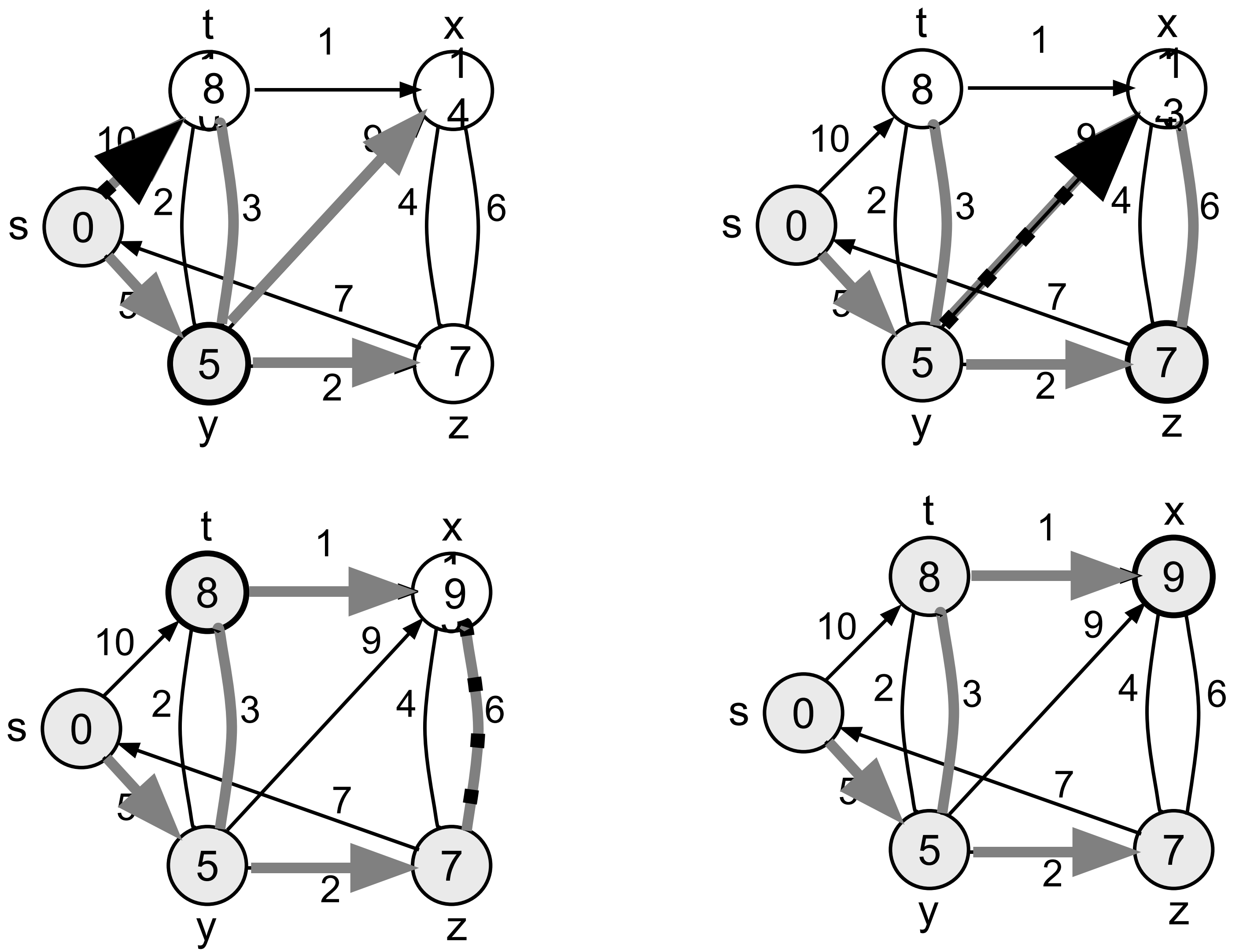
Dijkstra (G, w, s)

1. INITIALIZE-SINGLE-SOURCE(V, *s*)
2. S ← ∅
3. Q ← V[G]
4. **while** Q ≠ ∅
5. **do** u ← EXTRACT-MIN(Q)
6. S ← S ∪ {u}
7. **for** each vertex v ∈ *Adj*[u]
8. **do** RELAX(u, v, w)





Example



Dijkstra's Pseudo Code

- Graph *G*, weight function *w*, root *s*

```
DIJKSTRA(G, w, s)
for each v ∈ V
        do d[v] ← ∞
d[s] ← 0
S ← ∅    ▷ Set of discovered nodes
Q ← V
while Q ≠ ∅
        do u ← EXTRACT-MIN(Q)
           S ← S ∪ {u}
           for each v ∈ Adj[u]
                  do if d[v] > d[u] + w(u, v)
                        then d[v] ← d[u] + w(u, v)
```

relaxing edges

Dijkstra (G, w, s)

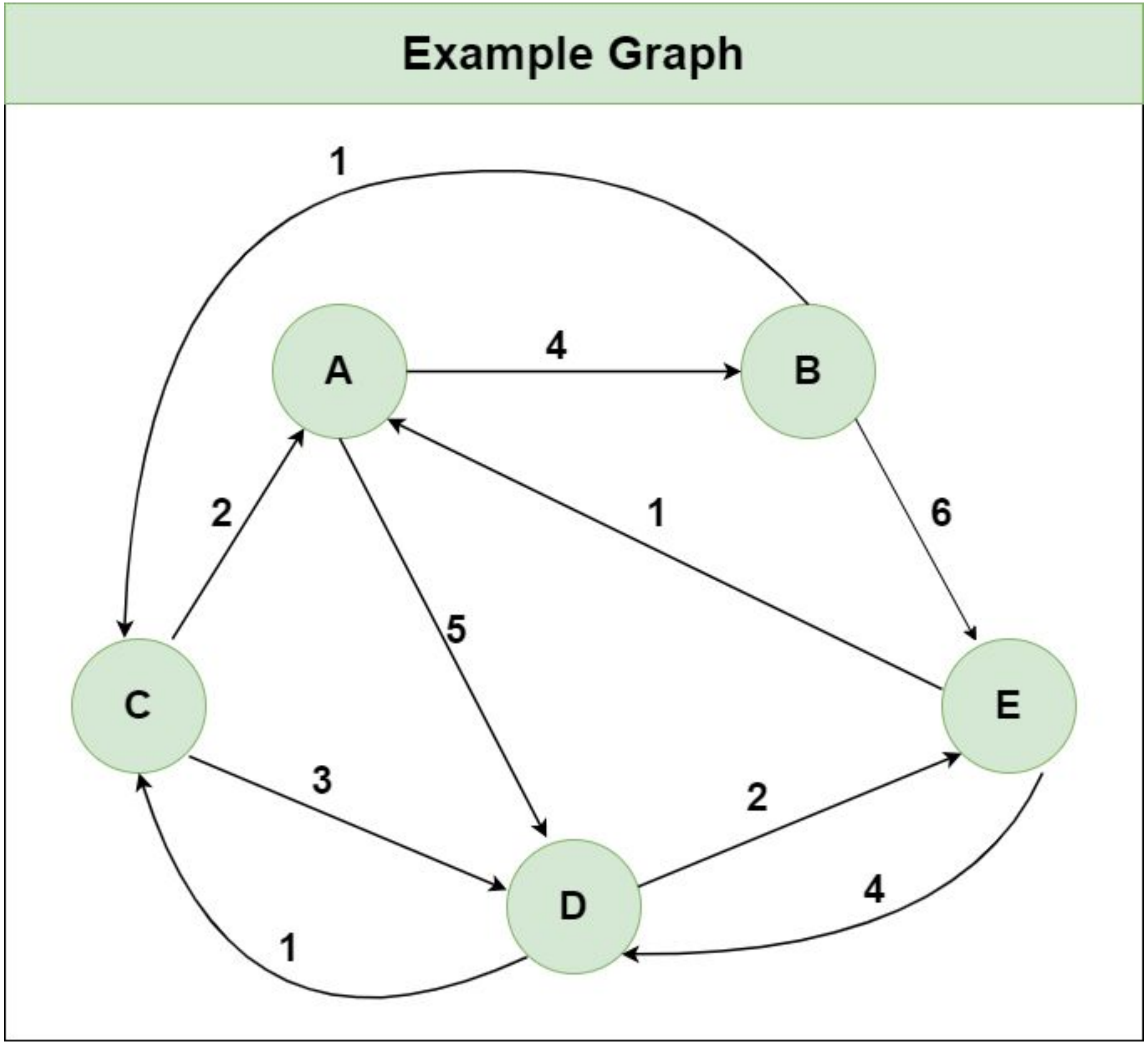
1. INITIALIZE-SINGLE-SOURCE(V, *s*) ⟵ Θ(V)
2. S ← ∅
3. Q ← V[G] ⟵ O(V) build min-heap
4. **while** Q ≠ ∅ ⟵ Executed O(V) times
5. **do** u ← EXTRACT-MIN(Q) ⟵ O(lgV)
6. S ← S ∪ {u}
7. **for** each vertex v ∈ Adj[u]
8. **do** RELAX(u, v, w) ⟵ O(E) times; O(lgV)

Running time: $O(V\lg V + E\lg V) = O(E\lg V)$

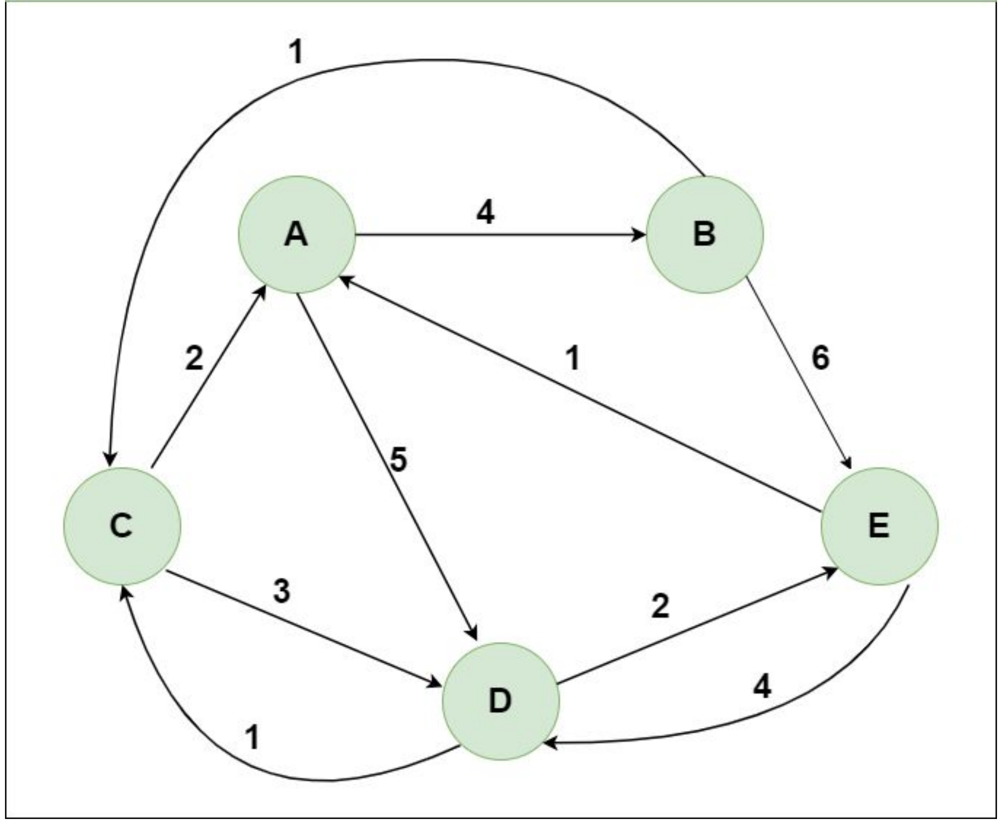
Dijkstra's Running Time

- Extract-Min executed $|V|$ time
- Decrease-Key executed $|E|$ time
- Time = $|V|\ T_{\text{Extract-Min}} + |E|\ T_{\text{Decrease-Key}}$
- T depends on different Q implementations

Q	T(Extract-Min)	T(Decrease-Key)	Total
array	$O(V)$	$O(1)$	$O(V^2)$
binary heap	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$
Fibonacci heap	$O(\lg V)$	$O(1)$ (amort.)	$O(V \lg V + E)$


Example Graph
1
A
4
B
2
1
6
5
C
E
3
2
D
4
1

Example Graph



Step1: Initializing Distance[ ][ ] using the Input Graph

	A	B	C	D	E
A	0	4	∞	5	∞
B	∞	0	1	∞	6
C	2	∞	0	3	∞
D	∞	∞	1	0	2
E	1	∞	∞	4	0

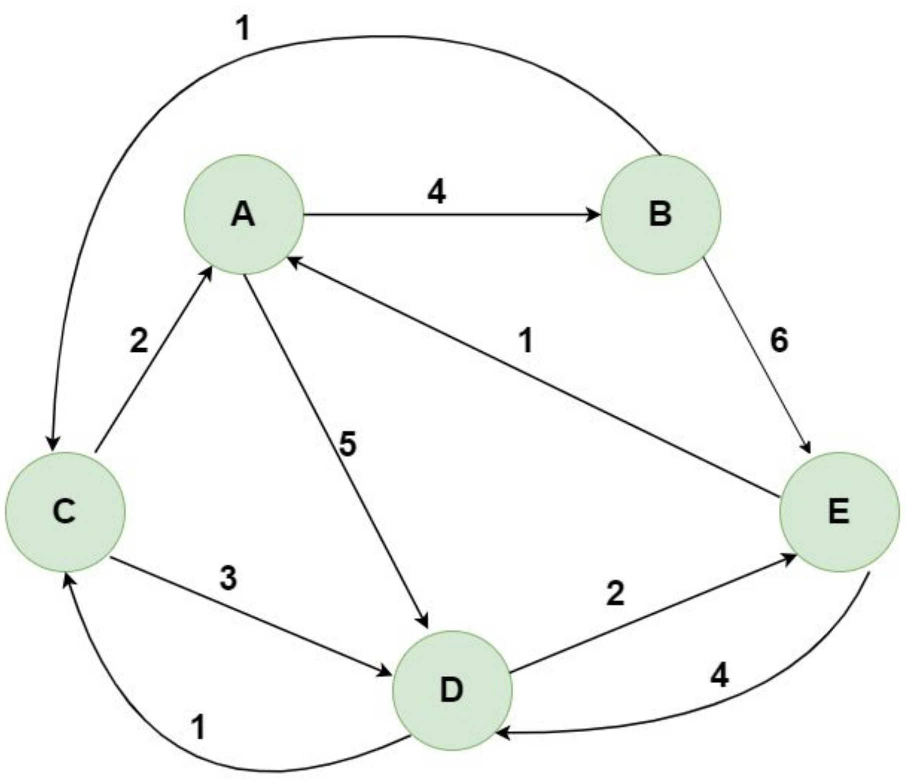
Step 2: Using Node A as the Intermediate node

Distance[i][j] = min (Distance[i][j], Distance[i][A] + Distance[A][j])

	A	B	C	D	E
A	0	4	∞	5	∞
B	∞	?	?	?	?
C	2	?	?	?	?
D	∞	?	?	?	?
E	1	?	?	?	?

	A	B	C	D	E
A	0	4	∞	5	∞
B	∞	0	1	∞	6
C	2	6	0	3	12
D	∞	∞	1	0	2
E	1	5	∞	4	0

Example Graph



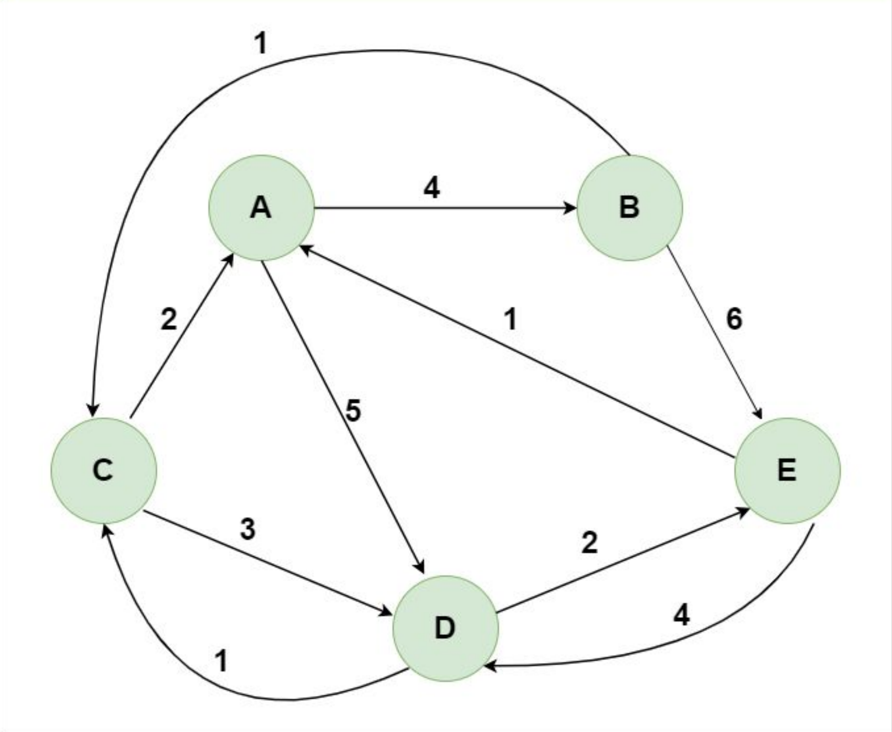
Step 3: Using Node B as the Intermediate node

Distance[i][j] = min (Distance[i][j], Distance[i][B] + Distance[B][j])

	A	B	C	D	E
A	?	4	?	?	?
B	∞	0	1	∞	6
C	?	6	?	?	?
D	?	∞	?	?	?
E	?	5	?	?	?

	A	B	C	D	E
A	0	4	5	5	10
B	∞	0	1	∞	6
C	2	6	0	3	12
D	∞	∞	1	0	2
E	1	5	6	4	0

Example Graph



Step 4: Using Node C as the Intermediate node

Distance[i][j] = min (Distance[i][j], Distance[i][C] + Distance[C][j])

	A	B	C	D	E
A	?	?	5	?	?
B	?	?	1	?	?
C	2	6	0	3	12
D	?	?	1	?	?
E	?	?	6	?	?

	A	B	C	D	E
A	0	4	5	5	10
B	3	0	1	4	6
C	2	6	0	3	12
D	3	7	1	0	2
E	1	5	6	4	0

Step 5: Using Node D as the Intermediate node

Distance[i][j] = min (Distance[i][j], Distance[i][D] + Distance[D][j])

	A	B	C	D	E
A	?	?	?	5	?
B	?	?	?	4	?
C	?	?	?	3	?
D	3	7	1	0	2
E	?	?	?	4	?

	A	B	C	D	E
A	0	4	5	5	7
B	3	0	1	4	6
C	2	6	0	3	5
D	3	7	1	0	2
E	1	5	5	4	0

Step 6: Using Node **E** as the Intermediate node

Distance[i][j] = min (Distance[i][j], Distance[i][E] + Distance[E][j])

	A	B	C	D	E
A	?	?	?	?	7
B	?	?	?	?	6
C	?	?	?	?	5
D	?	?	?	?	2
E	1	5	5	4	0

	A	B	C	D	E
A	0	4	5	5	7
B	3	0	1	4	6
C	2	6	0	3	5
D	3	7	1	0	2
E	1	5	5	4	0

Step 7: Return Distance[ ][ ] matrix as the result

	A	B	C	D	E
A	0	4	5	5	7
B	3	0	1	4	6
C	2	6	0	3	5
D	3	7	1	0	2
E	1	5	5	4	0

Textbooks & Web References

- Text Book (Chapter 24)
- www.geeksforgeeks.org

Thank you

&

Any question?